Efficient Computation of Buffer Capacities for Multi-Rate Real-Time Systems with Back-Pressure

Maarten Wiggers [1], Marco Bekooij [2] , Pierre Jansen [1], Gerard Smit [1]
[1] University of Twente, Enschede, The Netherlands
[2] Philips Research, Eindhoven, The Netherlands
m.h.wiggers@utwente.nl

ABSTRACT

A key step in the design of multi-rate real-time systems is the determination of buffer capacities. In our multi-processor system, we apply back-pressure as caused by bounded buffers in order to control jitter. This requires the derivation of buffer capacities that both satisfy the temporal constraints as well as constraints on the buffer capacity. Existing exact solutions suffer from the computational complexity associated with the required conversion from a multi-rate dataflow graph to a single-rate dataflow graph. In this paper we present an algorithm, with linear computational complexity, that does not require this conversion and that determines close to minimal buffer capacities. The algorithm is applied to an MP3 play-back application that is mapped on our network based multi-processor system.

Categories and Subject Descriptors: C.3 Special Purpose and application based systems Real-time and embedded systems

General Terms: Algorithms, Design, Performance

Keywords: System-on-Chip, Dataflow, Buffer Capacity

1. INTRODUCTION

Decreasing feature sizes have made it possible to implement multiple processing cores on a single chip, resulting in so-called Multi-Processor System-on-Chip (MPSoC) designs. These MPSoCs provide a high data processing throughput in a cost and energy-efficient way, making them an ideal match with multi-media applications as can be found in TV-sets, set-top boxes, and smart-phones.

MPSoCs operate on multiple streams of data that often have temporal constraints, such as throughput and latency. These streams have firm or soft real-time constraints. In the mentioned application domain, throughput constraints dominate over latency constraints.

For firm real-time streams we want to guarantee that no deadline is missed, because this would result in a severe quality degradation. Therefore, we model the processing performed on these streams with Multi-Rate Dataflow (MRDF) graphs [9], of which we can analytically derive the cycle that determines the throughput [13]. Tasks are modelled by the vertices of an MRDF graph, which are called actors.

An essential step when programming a multi-processor system is the determination of buffer capacities. In our multi-processor system, in which backpressure is applied, tasks start their execution on an assigned processor based on the availability of containers that signal the presence of data or space in a first-in first-out (FIFO) buffer with a fixed capacity. The buffer capacity therefore influences when tasks can start their execution and consequently influences the temporal behaviour of the stream. Applying back-pressure has the advantage that the system does not require means to control jitter, such as e.g. traffic shapers, to prevent buffer overflow.

However, determining whether particular buffer capacities allow a throughput that satisfies the constraint is a complex task. In order to find exact results, first a conversion from a multi-rate to a single-rate dataflow (SRDF) graph [9] is required, which can result in an exponential number of vertices [12], after which the throughput of each cycle in the SRDF graph is determined [3]. Algorithms that have a polynomial complexity for SRDF graphs, therefore have an exponential complexity for MRDF graphs.

The contribution of this paper is an algorithm that determines close to minimal buffer capacities that satisfy both the temporal constraint as well as any buffer capacity constraints that are for instance caused by finite memory sizes.

This is accomplished as follows. In our multi-processor system [1], we only use pre-emptive schedulers that provide resource budget guarantees [11]. These schedulers allow the determination of the response time of a task based on only the execution time of the task and the scheduler settings. This response time is therefore independent of other tasks. Using these response times, a strictly periodic schedule is constructed that meets the throughput constraint. Note that the effects of resource conflicts are already taken into account when constructing a schedule using response times. From this schedule we derive, in an analytical way, sufficient buffer capacities for the bounded FIFO buffers. This is possible, because our system has monotonic temporal behaviour, see Section 2.1, which guarantees that removing the constraint of a strictly periodic schedule at run-time will not lead to later container production times.

Alternative approaches make a different trade-off between complexity and accuracy. For instance, while we determine a one-dimensional periodic schedule, an alternative approach could include the determination of a multi-dimensional periodic schedule [14]. The number of dimensions equals the number of pairs of schedule start times and periods required

CODES+ISSS'06, October 22–25, 2006, Seoul, Korea

to describe the schedule. Multi-dimensional schedules can satisfy the throughput requirement and buffer capacity constraints for a larger set of problem instances, however determining such a schedule involves more complex computation.

Govindarajan [5] proposes a linear programming formulation that determines a schedule for each MRDF actor, with the corresponding number of SRDF actors as the number of dimensions, and aims at the minimum total buffer capacity required to satisfy the temporal constraints. This approach potentially leads to the minimally required total buffer capacity. However, because the objective function in their linear programming formulation is not exact, this goal is not always obtained. In our experience, this approach can lead to excessive run-times and memory requirements for realistic problem instances.

Our aim is to create a mapping flow that has as input an MRDF graph annotated with execution times together with the end-to-end throughput and latency constraints. The synthesis flow results in (1) the task to processor binding, (2) scheduler settings, and (3) buffer capacities such that our network-based multi-processor system [1] satisfies the temporal constraints. Such a mapping flow includes automatic design space exploration involving back-tracking, in which buffer capacities need to be determined that satisfy the constraints for multiple task to processor bindings and scheduler settings. The run-times of Govindarajan's approach can be problematic in case of automatic design space exploration involving back-tracking, in which buffer capacities need to be determined for multiple task to processor bindings and scheduler settings. However, in Section 7 we will show a realistic problem instance that caused the linear programming solver to run out of memory.

Scheduling approaches that do not include run-time arbitration, like the time triggered [8] and static-order [7] approaches, are difficult to apply for a system that includes a mix of streams that have firm or soft real-time constraints. This is because soft real-time streams often have execution times that are impractical to bound, and can have a data dependent number of task executions.

Scheduling approaches that include run-time arbitration, as for instance presented by Jersak [6], Goddard [4], or Maxiaguine [10], do not allow feedback cycles that influence the temporal behaviour of the system. Not only do these cycles occur, because of functional constraints, this also means that back-pressure through bounded FIFO buffers cannot be applied in these approaches.

The organization of this paper is as follows, some for this paper relevant properties of MRDF graphs are summerized in Section 2. The relation between an implementation and its MRDF graph is discussed in Section 3. In Section 4, we determine a proper schedule of two actors connected by a single edge, while in Section 5 upper bounds on the required number of tokens on a cycle are determined. Section 6 presents the algorithm to compute sufficient buffer capacities given an MRDF graph, response times and a throughput requirement. By applying the algorithm on an MP3 playback case study, we investigate the performance of the algorithm in Section 7.

2. ANALYSIS MODEL

The input to our mapping flow is an MRDF graph that models the application. An MRDF [9] graph is a directed graph $G = (V, E, \delta, \rho, \pi, \gamma)$ that consists of a finite set of actors V, and a set of directed edges, $E = \{(v_i, v_j) | v_i, v_j \in V\}$. Actors synchronise by communicating tokens over edges that represent queues. The graph G has an initial token placement $\delta : E \to \mathbb{N}$. An actor is enabled to fire when the number of tokens that will be consumed is available on all its input edges. The number of consumed tokens per firing is given by $\gamma : E \to \mathbb{N}$. Actor v_i therefore consumes $\gamma(e)$ tokens per firing from input edge $e = (v_k, v_i)$. The specified number of tokens is consumed in an atomic action from all input edges when the actor is started. The response time $\rho(v_i)$ is the difference between the finish and enabling time of an actor v_i, with $\rho : V \to \mathbb{R}$. When actor v_i finishes, then it produces the specified number of tokens on each output edge $e = (v_i, v_j)$ in one atomic action. The number of produced tokens per firing will be denoted by $\pi : E \to \mathbb{N}$.

We can describe the topology of an MRDF graph with a topology matrix Γ [9]. The matrix Γ is a $|E| \times |V|$ matrix, where

$$\Gamma_{ij} = \begin{cases} \pi(e_i) & \text{if } e_i = (v_j, v_k) \\ -\gamma(e_i) & \text{if } e_i = (v_k, v_j) \\ \pi(e_i) - \gamma(e_i) & \text{if } e_i = (v_j, v_j) \\ 0 & \text{otherwise} \end{cases}$$

If the rank of Γ is $|V| - 1$, then a connected MRDF graph is said to be consistent. A consistent MRDF graph requires queues with finite capacity, while an inconsistent MRDF graph requires infinite queue capacity. The vector $\boldsymbol{q}$ of length $|V|$, for which holds $\Gamma\mathbf{q} = \mathbf{0}$, is the repetition vector of the MRDF graph, which gives the relative firing frequencies of the actors.

For a strongly connected and consistent MRDF graph, we can specify a required period μ within which on average every actor v_x should fire q_x times. The throughput of the graph relates to μ^{-1}.

2.1 Monotonic Execution

If an MRDF graph is executed in a self-timed manner, then actors start execution as soon as they are enabled. Further we say that a FIFO ordering of tokens is maintained, if each actor either has a constant response time, or has a self-cycle with one initial token.

An important property is that self-timed execution of a strongly connected MRDF graph that maintains a FIFO ordering of tokens is monotonic in time, which is defined as follows.

DEFINITION 1 (MONOTONIC EXECUTION). *If the i'th firing of actor v_a consumes token j, then an MRDF graph executes monotonically if no decrease in response time of any firing of any actor can lead to a later enabling of the i'th firing of actor v_a.*

If an MRDF graph G maintains FIFO ordering of tokens, then the self-timed execution of G is monotonic. This is because a decrease in response time can only lead to earlier token production times, and therefore only to earlier actor enabling.

2.2 Example

An example MRDF graph is shown in Figure 1 in which data flows from actor v_1 to actor v_4. The repetition vector is $\mathbf{q} = [1\,2\,2\,4]^{\mathrm{T}}$, and the response times are $r_1 = 6$, $r_2 = 1$, $r_3 = 4$, and $r_4 = 2$. One instance of the problem discussed in this paper, is to find a token placement δ such that the period

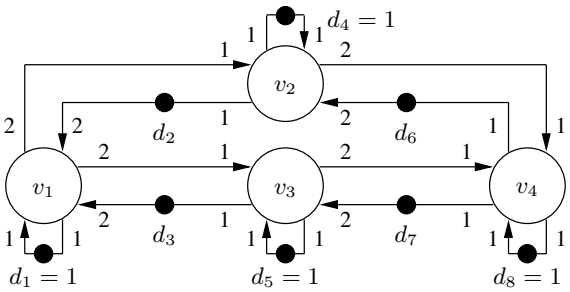

Figure 1: Example buffer capacity problem.

of the graph in Figure 1 is 8, and the constraints on d_1, d_4, d_5, and d_8 are satisfied. The presented algorithm will tell us that buffer capacities $d_1 = d_4 = d_5 = d_8 = 1$, $d_2 = 4$, $d_3 = 4$, $d_6 = 3$, and $d_7 = 4$ are sufficient to satisfy the constraints.

3. ANALYSIS MODEL AND IMPLEMENTATION

We assume the following. Our applications are implemented as a weakly connected directed task graph, of which the vertices represent tasks and the edges represent FIFO buffers with a fixed capacity. Tasks only communicate fixed sized containers over FIFO buffers, where a container can be full or empty. Further, a task communicates a fixed amount of containers per execution, and consumes containers after it has started and produces containers before it has finished. Furthermore, a task only starts when this fixed amount of containers is present on its input and output FIFO buffers. The finish time of each task execution is at most the worst case response time later than the enabling time. And at most one instance of a task can execute at any time.

The corresponding MRDF graph can now be constructed as follows. Each task is modelled by an actor that has a response time which equals the task's worst-case response time. Containers are represented by tokens. Each FIFO buffer is modelled by a pair of queues in opposite direction, where the tokens on one queue represent full containers, and on the other queue represent empty containers. The fixed capacity of a FIFO buffer is modelled by the number of initial tokens on the corresponding pair of queues. The fixed number of containers communicated per task execution is modelled by the actor token production and token consumption rates. The constraint that no two instances of a task can execute simultanuously is represented by an edge from the actor to itself, a self-cycle, with one initial token.

Since the task graph is weakly connected, and each FIFO buffer is represented by two queues in opposite direction the corresponding MRDF graph is strongly connected. And because each actor has a self-cycle with one initial token, the MRDF graph maintains a FIFO ordering of tokens.

In Section 2 we have established that self-timed execution of strongly connected MRDF graphs that maintain a FIFO ordering of tokens is monotonic in time. The MRDF graph is constructed in such a way that there is a one-to-one correspondence with the task graph, with the only important difference that tasks can have a smaller response time than actors and produce their containers before they finish. Therefore we arrive at the conclusion that, during self-timed execution, tasks produce their containers no later than the corresponding actors produce their tokens.

In the next section a strictly periodic schedule will be constructed for each actor. Compared to the strictly periodic schedule, the self-timed execution of the MRDF graph will not have later actor enabling times. This is because otherwise the strictly periodic schedule would violate a firing rule. Furthermore, the self-timed schedule in the implementation considers the actual response times. The task enabling times will therefore not be later than the corresponding actor enabling times. The buffer capacities derived using the strictly periodic schedule for the MRDF actors are therefore an upper bound on the buffer capacities that enable the self-timed schedule in the implementation to satisfy the throughput constraint.

4. SCHEDULE CONSTRUCTION

In this section strictly periodic schedules for two communicating actors are constructed such that the firing rules are not violated, the required temporal behaviour is satisfied, and a minimal buffer capacity is required.

We start by defining a number of variables that define properties of an edge and the actors connected to it. In this section we look at a single edge $e = (v_p, v_c)$ of a strongly connected and consistent MRDF graph. Edge e connects a token producing actor $v_p \in V$ to a token consuming actor $v_c \in V$, and has $d = \delta(e)$ initial tokens. Actor v_p has a response time $r_p = \rho(v_p)$, a token production rate $p = \pi(e)$, and a repetition factor q_p. Actor v_c has a response time $r_c = \rho(v_c)$, a token consumption rate $c = \gamma(e)$, and a repetition factor q_c.

We define the critical response time $\widehat{r}_x$ of actor v_x as $\widehat{r}_x = \frac{\mu}{q_x}$, where μ is the required period of the graph and q_x is the repetition rate of v_x.

4.1 Token production schedule

When considering an edge $e = (v_p, v_c)$, we construct a token production schedule such that v_p fires every $\widehat{r}_p$ time, where the first firing starts at $\widehat{r}_p - r_p$. Note that this schedule satisfies the required period μ. With this schedule the number of tokens produced is given by the function $\text{prod} : \mathbb{R} \rightarrow \mathbb{Z}$, $\text{prod}(t) = p\lfloor \frac{t}{\widehat{r}_p} \rfloor + d$.

4.2 Token consumption schedule

The token consumption schedule is constructed, similarly to the token production schedule, such that actor v_c fires every $\widehat{r}_c$ time. However, the token consumption schedule starts at $\beta + \widehat{r}_c - r_c$, where β is determined such that the token consumption schedule does not violate the firing rules. The number of tokens consumed by v_c is given by the function $\text{cons} : \mathbb{R} \rightarrow \mathbb{Z}$, $\text{cons}(t) = c\lfloor \frac{t-\beta+r_c}{\widehat{r}_c} \rfloor$. In the next paragraphs, we will determine an expression for β such that the required schedule of v_c is completely specified.

We first define $\text{prod}^{-1} : \mathbb{Z} \rightarrow \mathbb{R}$ and $\text{cons}^{-1} : \mathbb{Z} \rightarrow \mathbb{R}$ as follows, the production of token i occurs at time $\text{prod}^{-1}(i) = \widehat{r}_p \lceil \frac{i-d}{p} \rceil$, while the consumption of token i occurs at time $\text{cons}^{-1}(i) = \widehat{r}_c \lceil \frac{i}{c} \rceil + \beta - r_c$. The following lemmas will be helpful in the determination of β. Let gcd stand for greatest common divisor.

LEMMA 1. *For $l \in \mathbb{N}, 0 < l \leq \gcd(p, c)$ and $k = i * \gcd(p, c), i \in \mathbb{N}$, $\text{cons}^{-1}(k) - \text{prod}^{-1}(k) = \text{cons}^{-1}(k + l) - \text{prod}^{-1}(k + l)$.*

PROOF. From the edge $e = (v_p, v_c)$ with production rate p and consumption rate c, we can create a new edge e' where

the token size is multiplied with $\gcd(p,c)$, the number of tokens is divided by $\gcd(p,c)$ and with production rate $\frac{p}{\gcd(p,c)}$ and consumption rate $\frac{c}{\gcd(p,c)}$. The token production and consumption events on e' are at the same times as the token production and consumption events on e. This is because p and c are scaled by the same factor resulting in $c' = \frac{c}{\gcd(p,c)}$ and $p' = \frac{p}{\gcd(p,c)}$, where $p', c' \in \mathbb{N}$. □

LEMMA 2. *For every $x, y \in \mathbb{Z}$ it holds that $\lceil \frac{x}{y} \rceil \leq \frac{x}{y} + 1 - \frac{\gcd(x,y)}{y}$.*

PROOF. If $a, b \in \mathbb{Z}$ and $\gcd(a,b) = 1$, then $\lceil \frac{a}{b} \rceil \leq \frac{a}{b} + 1 - \frac{1}{b}$. Therefore if $s, t \in \mathbb{Z}$ and $\gcd(s,t) = d$, then $\lceil \frac{s}{t} \rceil = \lceil \frac{s/d}{t/d} \rceil \leq \frac{s/d}{t/d} + 1 - \frac{1}{t/d} = \frac{s}{t} + 1 - \frac{\gcd(s,t)}{t}$. □

LEMMA 3. *For every $x, y \in \mathbb{Z}$ it holds that $\lfloor \frac{x}{y} \rfloor \geq \frac{x}{y} - 1 + \frac{\gcd(x,y)}{y}$.*

PROOF. If $a, b \in \mathbb{Z}$ and $\gcd(a,b) = 1$, then $\lfloor \frac{a}{b} \rfloor \geq \frac{a}{b} - 1 + \frac{1}{b}$. Therefore if $s, t \in \mathbb{Z}$ and $\gcd(s,t) = d$, then $\lfloor \frac{s}{t} \rfloor = \lfloor \frac{s/d}{t/d} \rfloor \geq \frac{s/d}{t/d} - 1 + \frac{1}{t/d} = \frac{s}{t} - 1 + \frac{\gcd(s,t)}{t}$. □

LEMMA 4. *If in a consistent MRDF graph, actors v_1 and v_2 are connected by an edge $e = (v_1, v_2)$, then $\frac{\widehat{r}_1}{\pi(e)} = \frac{\widehat{r}_2}{\gamma(e)}$.*

PROOF. By construction we have $\mu = \widehat{r}_1 q_1 = \widehat{r}_2 q_2 \rightarrow \frac{\widehat{r}_1}{\widehat{r}_2} = \frac{q_2}{q_1}$. Since the MRDF graph is consistent we have that $q_1 \pi(e) = q_2 \gamma(e) \rightarrow \frac{q_2}{q_1} = \frac{\pi(e)}{\gamma(e)}$, resulting in $\frac{\widehat{r}_1}{\widehat{r}_2} = \frac{\pi(e)}{\gamma(e)}$. □

THEOREM 1. *A sufficient start value $\beta \in \mathbb{R}$ of the strictly periodic schedule of the token consuming actor v_c such that the firing rule is not violated is $r_c + \widehat{r}_p - \frac{\widehat{r}_p \gcd(p,c)}{p} - \gcd(p,c) \lfloor \frac{d}{\gcd(p,c)} \rfloor \frac{\widehat{r}_p}{p}$.*

PROOF. The difference between cons^{-1} and prod^{-1} only changes every $\gcd(p,c)$ tokens, see Lemma 1. Therefore instead of considering $\text{prod}^{-1}(i) = \widehat{r}_p \lceil \frac{i-d}{p} \rceil, i \in \mathbb{N}$, we can consider $\text{prod}^{-1}(k) = \widehat{r}_p \lceil \frac{k-d}{p} \rceil$, with $k = i\gcd(p,c)$. From Lemma 2 we know that $\widehat{r}_p \lceil \frac{k-d}{p} \rceil \leq \widehat{r}_p (\frac{k-d}{p} + 1 - \frac{\gcd(k-d,p)}{p})$. If we define $\lambda = j\gcd(p,c)$, with $j = \lfloor \frac{d}{\gcd(p,c)} \rfloor, j \in \mathbb{N}$, then $\gcd(k-\lambda, p) = \gcd(i\gcd(p,c) - j\gcd(p,c), p) \geq \gcd(p,c)$ for $i - j \geq 1$, we have that $\frac{\widehat{r}_p(k-\lambda)}{p} + \widehat{r}_p - \frac{\widehat{r}_p \gcd(k-\lambda,p)}{p} \leq \frac{\widehat{r}_p(k-\lambda)}{p} + \widehat{r}_p - \frac{\widehat{r}_p \gcd(p,c)}{p}$ again for $i - j \geq 1$. For $i - j = 0$ and thus $k - \lambda = 0$, we have that $\widehat{r}_p \lceil \frac{k-\lambda}{p} \rceil = 0$ while $\frac{\widehat{r}_p(k-\lambda)}{p} + \widehat{r}_p - \frac{\widehat{r}_p \gcd(p,c)}{p} = \widehat{r}_p - \frac{\widehat{r}_p \gcd(p,c)}{p} = \widehat{r}_p (1 - \frac{\gcd(p,c)}{p})$. Since $\gcd(p,c) \leq p$ we have that $\widehat{r}_p (1 - \frac{\gcd(p,c)}{p}) \geq 0$. Therefore $\text{prod}^{-1}(k) \leq \frac{\widehat{r}_p(k-\lambda)}{p} + \widehat{r}_p - \frac{\widehat{r}_p \gcd(p,c)}{p}$.

Further we know that $\text{cons}^{-1}(k) = \widehat{r}_c \lceil \frac{k}{c} \rceil + \beta - r_c \geq \frac{\widehat{r}_c k}{c} + \beta - r_c$.

Thus a lower bound on the difference between cons^{-1} and prod^{-1} is $(\frac{\widehat{r}_c k}{c} + \beta - r_c) - (\frac{\widehat{r}_p(k-\lambda)}{p} + \widehat{r}_p - \frac{\widehat{r}_p \gcd(p,c)}{p})$, since consumption of token k cannot occur before production of token k, we obtain the following: $\frac{\widehat{r}_c k}{c} + \beta - r_c - \frac{\widehat{r}_p(k-\lambda)}{p} - \widehat{r}_p + \frac{\widehat{r}_p \gcd(p,c)}{p} \geq 0$. Using Lemma 4 we know $\frac{\widehat{r}_p}{p} = \frac{\widehat{r}_c}{c}$. The inequality for the minimum difference therefore reduces to $\beta - r_c + \frac{\lambda \widehat{r}_p}{p} - \widehat{r}_p + \frac{\widehat{r}_p \gcd(p,c)}{p} \geq 0 \rightarrow \beta \geq r_c + \widehat{r}_p - \frac{\widehat{r}_p \gcd(p,c)}{p} - \lambda \frac{\widehat{r}_p}{p} = r_c + \widehat{r}_p - \frac{\widehat{r}_p \gcd(p,c)}{p} - \gcd(p,c) \lfloor \frac{d}{\gcd(p,c)} \rfloor \frac{\widehat{r}_p}{p}$. □

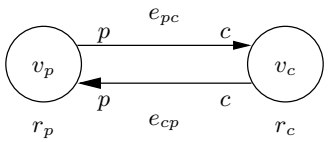

Figure 2: The cycle considered in this section.

5. BUFFER CAPACITY

In this section we will determine the required buffer capacity of a FIFO buffer modelled with a cycle C through the two actors v_p and v_c. The cycle C is formed by the edges $e_{pc} = (v_p, v_c)$ and $e_{cp} = (v_c, v_p)$, see Figure 2. The edge e_{pc} determines a difference $\alpha \in \mathbb{R}$ between the start times of the schedules of v_p and v_c, while tokens can be placed on edge e_{cp}.

While $\text{prod}(t)$ and $\text{cons}(t)$ are defined on the edge e_{pc}, we define the consumption of tokens by v_p on edge $e_{cp} = (v_c, v_p)$ with acq: $\mathbb{R} \rightarrow \mathbb{Z}$, $\text{acq}(t) = p \lfloor \frac{t+r_p}{\widehat{r}_p} \rfloor$. As required by the model, consumption of tokens by v_p on e_{cp} occurs r_p earlier than production of tokens on e_{pc}. Production of tokens on e_{cp} by v_c occurs when v_c finishes, and is thus given by rel: $\mathbb{R} \rightarrow \mathbb{Z}$, $\text{rel}(t) = c \lfloor \frac{t-\alpha}{\widehat{r}_c} \rfloor$. Further $\text{acq}^{-1} : \mathbb{Z} \rightarrow \mathbb{R}$, $\text{acq}^{-1}(i) = \widehat{r}_p \lceil \frac{i}{p} \rceil - r_p$ and $\text{rel}^{-1} : \mathbb{Z} \rightarrow \mathbb{R}$ with $\text{rel}^{-1}(i) = \widehat{r}_c \lceil \frac{i}{c} \rceil + \alpha$.

The required number of tokens on the cycle C can be found by determining the maximum difference between the number of tokens acquired by v_p, $\text{acq}(t)$,and the number of tokens released by v_c, $\text{rel}(t)$, that will ever occur when v_p and v_c execute according to the constructed schedules.

THEOREM 2. *A sufficient number of tokens to be placed on edge e_{cp} to meet the required period μ equals $\gcd(p,c) \lfloor \frac{\frac{p(r_p+\alpha)}{\widehat{r}_p} + c}{\gcd(p,c)} \rfloor$.*

PROOF. We will first derive an expression that forms a lower bound on $\text{rel}(t)$, and subsequently an upper bound on $\text{acq}(t)$, since $\text{acq}(t) \geq \text{rel}(t)$ we find an upper bound on the difference between $\text{acq}(t)$ and $\text{rel}(t)$ by subtracting these bounds.

We know that $\text{rel}(t) = c \lfloor \frac{t-\alpha}{\widehat{r}_c} \rfloor \geq \frac{c(t-\alpha)}{\widehat{r}_c} - c$. We further know that $\text{acq}(t) = p \lfloor \frac{t+r_p}{\widehat{r}_p} \rfloor \leq p \frac{t+r_p}{\widehat{r}_p}$. An upper bound on $\text{acq}(t) - \text{rel}(t)$ is therefore $(p \frac{t+r_p}{\widehat{r}_p}) - (c \frac{t-\alpha}{\widehat{r}_c} - c) = \frac{pt}{\widehat{r}_p} + \frac{pr_p}{\widehat{r}_p} - \frac{ct}{\widehat{r}_c} + \frac{c\alpha}{\widehat{r}_c} + c$.

From Lemma 4 we obtain $\frac{\widehat{r}_c}{c} = \frac{\widehat{r}_p}{p} \rightarrow \frac{p}{\widehat{r}_p} = \frac{c}{\widehat{r}_c}$. The expression for the maximum difference between $\text{acq}(t)$ and $\text{rel}(t)$ thus reduces to $\frac{pr_p}{\widehat{r}_p} + \frac{c\alpha}{\widehat{r}_c} + c = \frac{pr_p}{\widehat{r}_p} + \frac{p\alpha}{\widehat{r}_p} + c = \frac{p(r_p+\alpha)}{\widehat{r}_p} + c$.

Since the difference d between the number of tokens placed on and removed from edge e_{cp} is a linear combination of p and c, d is a multiple of $\gcd(p,c)$. And since the obtained upper bound on the maximum difference is not necessarily a multiple of $\gcd(p,c)$ we can round to the next smaller integer that is a multiple of $\gcd(p,c)$, obtaining $\gcd(p,c) \lfloor \frac{\frac{p(r_p+\alpha)}{\widehat{r}_p} + c}{\gcd(p,c)} \rfloor$. □

If, however, $\widehat{r}_p, \widehat{r}_c \in \mathbb{N}$, then the bound on the maximum difference between $\text{acq}(t)$ and $\text{rel}(t)$ as obtained in the proof of Theorem 2 can be improved. Note that if $\widehat{r}_p, \widehat{r}_c \in \mathbb{Q}$, then multiplying $\widehat{r}_p$ and $\widehat{r}_c$ by the least common multiple of their denominators results in integer values. And for the sake of completeness, if $\widehat{r}_p, \widehat{r}_c \in \mathbb{R}$, then rounding to a next

larger value in $\mathbb{Q}$ is possible in order to derive sufficient buffer capacities, due to the monotonic behaviour of our system.

THEOREM 3. *If $\widehat{r}_p, \widehat{r}_c \in \mathbb{N}$, then a sufficient number of tokens to be placed on edge e_{cp} to meet the required period μ equals $gcd(p,c)\lfloor \frac{\frac{p(r_p+\alpha-1)}{\widehat{r}_p}+c}{\gcd(p,c)} \rfloor$.*

PROOF. If time is discrete, Lemma 3 can be applied to obtain $\text{rel}(t) = c\lfloor \frac{t-\alpha}{\widehat{r}_c} \rfloor \geq c(\frac{t-\alpha}{\widehat{r}_c} - 1 + \frac{1}{\widehat{r}_c}) = c\frac{t-\alpha+1}{\widehat{r}_c} - c$.

An upper bound on $\text{acq}(t) - \text{rel}(t)$ is therefore $(p\frac{t+r_p}{\widehat{r}_p}) - (c\frac{t-\alpha+1}{\widehat{r}_c} - c) = \frac{pt}{\widehat{r}_p} + \frac{pr_p}{\widehat{r}_p} - \frac{ct}{\widehat{r}_c} + \frac{c\alpha}{\widehat{r}_c} - \frac{c}{\widehat{r}_c} + c$.

From Lemma 4 we obtain $\frac{\widehat{r}_c}{c} = \frac{\widehat{r}_p}{p} \rightarrow \frac{p}{\widehat{r}_p} = \frac{c}{\widehat{r}_c}$. The expression for the maximum difference between $\text{acq}(t)$ and $\text{rel}(t)$ thus reduces to $\frac{pr_p}{\widehat{r}_p} + \frac{c\alpha}{\widehat{r}_c} - \frac{c}{\widehat{r}_c} + c = \frac{pr_p}{\widehat{r}_p} + \frac{p\alpha}{\widehat{r}_p} - \frac{p}{\widehat{r}_p} + c = \frac{p(r_p+\alpha-1)}{\widehat{r}_p} + c$.

Since the difference d between the number of tokens placed on and removed from edge e_{cp} is a linear combination of p and c, d is a multiple of $\gcd(p,c)$. And since the obtained upper bound on the maximum difference is not necessarily a multiple of $\gcd(p,c)$ we can round to the next smaller integer that is a multiple of $\gcd(p,c)$, obtaining $\gcd(p,c)\lfloor \frac{\frac{p(r_p+\alpha-1)}{\widehat{r}_p}+c}{\gcd(p,c)} \rfloor$. □

6. ALGORITHM

In this section we present the algorithm to compute sufficient buffer capacities of a strongly connected and consistent MRDF graph $G = (V, E, \delta, \rho, \pi, \gamma)$, given a repetition vector $\mathbf{q}$, and a required period μ. The presented algorithm is applicable if the dependencies caused by the communication over bounded FIFO buffers are the only dependencies between tasks, and if each bounded FIFO buffer is either full or empty. Therefore, apart from any self-cycles, the corresponding MRDF graph only has pairs of edges between actors, with each pair representing a bounded FIFO buffer. On each such pair of edges, all initial tokens are assumed to be on a single edge, which models that a bounded FIFO buffer is initially either full or empty. The set of edges on which tokens can be placed forms a set $B \subset E$, and $\overline{B} = E \setminus B$. The initial tokens $\rho(e)$ on each edge e are considered as the constraint on the buffer capacity. In the following algorithm, the schedule of each actor v_i will be assigned an as-soon-as-possible $\text{asap}(v_i)$ and an as-late-as-possible $\text{alap}(v_i)$ start time, such that the constraints are satisfied and the required buffer capacity is close to minimal.

1. $\forall v_i \in V \bullet \widehat{r}_i = \frac{\mu}{q_i}$, $\text{asap}(v_i) = 0$, and $\text{alap}(v_i) = \infty$
2. Determine β_{ij} for each edge $(v_i, v_j) \in E$
3. Create a directed a-cyclic graph $\overline{D} = (V, \overline{B}, \delta, \rho, \pi, \gamma)$
4. Create a list L of length $|V|$, where the actors of $\overline{D}$ are topologically sorted. If the topological sort fails, then report deadlock
5. Visit the actors in L from front to back, and for each actor v_i : $\text{asap}(v_i) = \max(\{\text{asap}(v_i)\} \cup \{\text{asap}(v_x) + \beta_{xi} | (v_x, v_i) \in \overline{B}\})$
6. $\forall v_l \in V \setminus \{v_x | \exists v_y \bullet (v_x, v_y) \in \overline{B}\} \bullet \text{alap}(v_l) = \text{asap}(v_l)$
7. Visit the actors in L from back to front, and for each actor v_i : $\text{alap}(v_i) = \min(\{\text{alap}(v_i)\} \cup \{\text{alap}(v_x) - \beta_{ix} | (v_i, v_x) \in \overline{B}\})$
8. Visit the actors in L from front to back, and for each actor v_i : $\text{alap}(v_i) = \min(\{\text{alap}(v_i)\} \cup \{\text{alap}(v_x) - \beta_{ix} | (v_i, v_x) \in B\})$, and report a constraint violation if $\text{asap}(v_i) > \text{alap}(v_i)$
9. For each edge $(v_j, v_i) \in B$, a sufficient number of tokens can be determined using Theorem 3, if we take $\alpha = \text{alap}(v_j) - \text{alap}(v_i)$.

Step 3 creates a connected directed a-cyclic graph. The graph $\overline{D}$ is connected, because between each communicating pair of actors v_x and v_y there is at least one edge on which no tokens can be placed: $(v_x, v_y) \in \overline{B} \vee (v_y, v_x) \in \overline{B}$. And further $\overline{D}$ is a-cyclic, because it is not possible that there remains a cycle, since this would indicate deadlock [13].

For the following reason, all actors in V will have been assigned alap times after step 7. Since $\overline{D}$ is a connected graph, all actors without successors in $\overline{D}$ have been assigned an asap time after step 5, and thus after step 6 all actors without successors in $\overline{D}$ have been assigned an alap time. If we would create a graph $D = (V, B, \delta, \rho, \pi, \gamma)$, then the actors without successors in $\overline{D}$ are the actors without predecessors in D. After step 6, the actors without predecessors in D have all been assigned alap times, while in step 7 all actors reachable from an actor with an alap time will be assigned an alap time. And we have that the union of the sets of actors reachable from the actors without predessors in D equals V.

The three passes through L are required for the following reasons. In step 5 each actor is assigned the minimal asap time that does not violate the constraints formed by edges from $\overline{B}$, given that all vertices with no predecessors in $\overline{D}$ have an asap time that equals 0. In step 7 each actor is assigned the maximal alap time that does not violate the constraints formed by edges from $\overline{B}$, given that all vertices with no successors in $\overline{D}$ have an alap time that equals their asap time. In step 8 each actor is assigned the maximal alap time that does not violate the constraints formed by edges from B, the maximal alap time is taken to impose as few restrictions as possible on alap times that are determined later. Subsequently a constraint feasibility check is performed.

In the MRDF graph of Figure 1, the asap and alap times of actor v_2 are different, because $r_3 > r_2$ results in $\beta_{34} > \beta_{24}$. In this example the difference between the asap and alap times is large enough to make a trade-off between the buffer capacities d_2 and d_6, i.e. $d_2 = 3$ and $d_6 = 4$ are also sufficient buffer capacities.

Note that, because the presented algorithm constructs, in general non-optimal, strictly periodic schedules, it can occur that an in fact feasible constraint set is reported infeasible by this algorithm. Furthermore, note that even though this algorithm determines buffer capacities such that the constructed schedule does not experience back-pressure, the self-timed schedule in the implementation will experience back-pressure.

Steps 1, 5, 6, and 7 each have complexity $O(V)$, steps 3 and 4 have complexity $O(V + E)$ [2], and steps 2, 8, and 9 have complexity $O(E)$. The complexity of this algorithm is therefore $O(V+E)$. This low complexity is due to the specific class of MRDF graphs that this algorithm can deal with. In general a single source longest path algorithm, as for instance the Bellman-Ford algorithm with complexity $O(VE)$ [2], is required to determine asap and alap times [13].

7. EXPERIMENTAL RESULTS

In this section we apply our algorithm to determine buffer capacities for an MP3 playback application. In the MP3 playback application, of which the MRDF graph is shown in

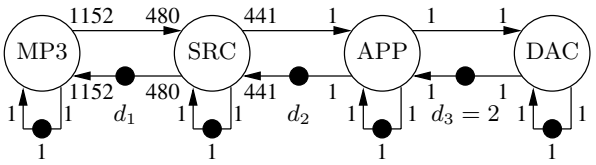

Figure 3: MP3 playback application.

Figure 3, the compressed audio is decoded by the MP3 task into a 48kHz audio sample stream. These samples are converted by the Sample Rate Converter (SRC) task into a 44.1 kHz stream, after which the Audio Post-Processing (APP) task enhances the perceived quality of the audio stream and sends the samples to a Digital to Analog Converter (DAC).

We performed an experiment with the MP3 playback application in which the topology of the MRDF graph is fixed to the topology shown in Figure 3, the required period is fixed to $\mu = 1217160$, and the response time of the sample rate converter is varied to show the behaviour of the presented algorithm. The repetition vector is $[5\ 12\ 5292\ 5292]^{\mathrm{T}}$, further $r_{MP3} = \widehat{r}_{MP3}$, $r_{APP} = \widehat{r}_{APP}$, and $r_{DAC} = \widehat{r}_{DAC}$.

Table 1 lists the resulting buffer sizes for this experiment, both as determined by the algorithm presented in Section 6 and as determined through back-tracking where the throughput of the MRDF graph is determined with Maximum Cycle Mean (MCM) analysis [13], which is applied on the SRDF graph.

r_{SRC}	Buffer Capacity					
	d_1		d_2		$d_1 + d_2$	
	alg.	opt.	alg.	opt.	alg.	opt.
101430	3072	3072	882	882	3954	3954
76073	2976	2688	772	1015	3748	3703
50715	2880	2688	662	794	3542	3482
25358	2784	2688	552	574	3336	3262

Table 1: Results from presented algorithm (alg.) and the optimal result (opt.) for MP3 playback.

The presented algorithm confirms that one token on each self-cycle and $d_3 = 2$ is sufficient. Further, as shown in Table 1, the algorithm has a reasonable accuracy at a run-time in the order of 10^{-2}s. In our multi-processor system, often more than one FIFO buffer will be mapped on a particular memory. This results in a constraint on a sum of buffer capacities. Since the algorithm does not allow for constraints that include multiple buffer capacities, many iterations using this algorithm are required. The low run-time of a single iteration enables this approach.

Even though we have been able to apply back-tracking in combination with MCM analysis for this example, we feel that this is not a feasible approach since one iteration of the Howard algorithm [3], which we used to derive the MCM, requires a minute.

Application of Govindarajan's linear programming formulation on the MP3 playback application was infeasible, because the solver from the GNU Linear Programming Kit (GLPK) runs out of memory. Removal of the APP task from the graph enables the application of Govindarajan's approach, but still has a run-time of half an hour.

Note that there exists a trade-off between the buffer capacities, similar to the trade-off in the MRDF graph of Figure 1, that is not apparent in the MRDF graph of Figure 3, but which is required to be exploited in order to obtain minimal buffer capacities, see Table 1. Conversion of the MRDF graph into an SRDF graph is required to make this trade-off explicit.

8. CONCLUSION

In this work we have presented an algorithm that determines close to minimal buffer capacities for Multi-Rate Dataflow graphs such that the throughput requirement and constraints on buffer capacities are satisfied. The buffer capacities are analytically determined from a constructed strictly periodic schedule. Because this algorithm does not require a conversion from a Multi-Rate Dataflow graph to a Single-Rate Dataflow graph, we do not suffer from the exponential complexity associated with this conversion. Related algorithms do make this conversion and have excessive run-times and memory requirements for realistic Multi-Rate Dataflow graphs.

An essential difference with related real-time synthesis approaches is that functional and flow control cycles are allowed to influence the temporal behaviour. In our system, jitter can therefore be controlled through back-pressure implemented by bounded FIFO buffers.

We are currently setting up a mapping flow that determines both scheduler settings, which result in appropriate response times, and the task to processor assignment. In order to derive a configuration that meets all constraints, we expect that this mapping flow will need to evaluate many different scheduler settings and task to processor assignments, for which the presented algorithm will be an important contribution.

9. REFERENCES

[1] M. Bekooij *et al. Dataflow Analysis for Real-Time Embedded Multiprocessor System Design*, chapter 15. Dynamic and Robust Streaming Between Connected CE Devices. Kluwer Academic Publishers, 2005.

[2] T. Cormen *et al. Introduction to Algorithms*. MIT press, 2001.

[3] A. Dasdan. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems*, 9(4):385–418, October 2004.

[4] S. Goddard *et al.* Managing latency and buffer requirements in processing graph chains. *The Computer Journal*, 44(6), 2001.

[5] R. Govindarajan *et al.* Minimizing buffer requirement under rate-optimal schedules in regular dataflow networks. *Journal of VLSI Signal Processing*, 31(3), 2002.

[6] M. Jersak *et al.* Performance analysis of complex embedded systems. *International Journal of Embedded Systems*, 1(1-2):33–49, 2005.

[7] V. Kianzad *et al.* An architecturral level design methodology for embedded face detection. In *Proceedings of CODES+ISSS'05*, pages 136–141, September 2005.

[8] H. Kopetz. *Real-Time Systems: Design Principles for Distibuted Embedded Applications*. Kluwer Academic Publishers, 1997.

[9] E. A. Lee *et al.* Synchronous dataflow. *Proceedings of the IEEE*, 75(9):1235–1245, September 1987.

[10] A. Maxiaguine *et al.* Tuning soc platforms for multimedia processing: Identifying limits and tradeoffs. In *Proceedings of CODES+ISSS'04*, pages 128–133, Stockholm, September 2004.

[11] C. W. Mercer *et al.* Processor capacity reserves: Operating system support for multimedia applications. In *Proceedings ICMCS*, pages 90–99, 1994.

[12] J. L. Pino *et al.* Hierarchical static scheduling of dataflow graphs onto multiple processors. In *Proceedings of ICASSP*, May 1995.

[13] S. Sriram *et al. Embedded Multiprocessors: Scheduling and Synchronization*. Marcel Dekker Inc., 2000.

[14] W. F. J. Verhaegh *et al.* A two-stage solution approach to multidimensional periodic scheduling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(10):1185–1199, October 2001.